Enforcing Service Availability in Mobile Ad-Hoc WANs[*]

Levente Buttyán and Jean-Pierre Hubaux
Institute for computer Communications and Applications
Communication Systems Department
Swiss Federal Institute of Technology
EPFL-DSC-ICA, CH-1015 Lausanne, Switzerland
{Levente.Buttyan, Jean-Pierre.Hubaux}@epfl.ch

3 May 2000

Abstract

In this paper, we address the problem of service availability in mobile ad-hoc WANs. We present a secure mechanism to stimulate end users to keep their devices turned on, to refrain from overloading the network, and to thwart tampering aimed at converting the device into a "selfish" one. Our solution is based on the application of a tamper resistant security module in each device and cryptographic protection of messages.

1 Introduction

1.1 The context

The Terminodes Project [1, 2] is a 10-year research program (2000-2010) with the aim to investigate wide area, large, totally wireless, mobile networks that we call *mobile ad-hoc wide area networks*. In this project, we follow a radically distributed approach, in which all networking functions are embedded in the terminals themselves. Because they act as network nodes and terminals at the same time, we call these devices *terminodes*. A network of terminodes is an autonomous, self-organized network, completely independent of any fixed infrastructure or other equipment.

Our vision of the Terminodes Project can be illustrated by a free, amateur, wireless ad-hoc network covering a wide area, which operates at unlicensed frequencies. In this scenario, terminodes are small personal devices that everyone in the area could potentially own. The size of the network can reach several million devices in regions of high density population. Communication among users is based on packet switched[1], multi-hop, wireless communication of voice and data. An important characteristic of terminode networks is that there are no routing tables stored in the devices. Instead, a simple *packet forwarding* mechanism lets each of the terminodes located on the route of a given packet compute the "best" next hop toward the final destination [3].

[*]Technical Report No. DSC/2000/025

[1]While circuit switching is an advantage for supporting voice, the complexity associated with establishing, maintaining, and releasing circuits, or any form of connection, is at odds with the requirement that intermediate systems are user equipment, and may operate quite irregularly.

1.2 The problem

The problem that we address in this paper is the availability of services in terminode networks. In civilian applications of ad-hoc networks, which we are exclusively concerned with in the Terminodes Project, availability is often considered to be the security issue of greatest relevance for users [4]. We concentrate on two aspects of availability in terminode networks:

- **Stimulation for co-operation.** Since *all* networking services (e.g., packet forwarding, mobility management) should be provided by the terminodes themselves, these services are available only if the terminodes (or, more precisely, their users) are willing to provide them. On the other hand, service provision is not in the direct interest of users, because it consumes energy and thus, reduces battery lifetime. Therefore, a stimulation mechanism that encourages users to leave their terminodes switched on and let them provide services to other terminodes is required.

  One can say that being able to receive messages is enough motivation for the user to leave her terminode switched on. While this may indeed be true, it is certainly not enough to encourage users to provide services to other terminodes. The hardware and the software of the terminode can be tampered with and their behavior can be modified by the user in a way that the device can receive messages but it does not provide any services to the community. Furthermore, criminal organizations can tamper with terminodes and sell corrupted devices, which do not co-operate in order to save energy, on a large scale.

  So far, civilian applications of ad-hoc networks have been envisioned mainly in crisis situations (e.g., rescue operations). For this reason, it was assumed that users are naturally motivated to co-operate. In terminode networks, this assumption does not hold, because of the size of the network, and because we consider that the network lifetime can be long (typically, several years).

- **Prevention of overloading.** Often, services are unavailable because the network is overloaded and it can no longer carry useful information. The network can become overloaded because of a malicious denial-of-service attack, or simply because some of the (otherwise legitimate) users want to send too much information. Therefore, we need a mechanism that makes denial-of-service attacks "expensive" and discourages users from flooding the network with useless traffic. In cellular networks, this objective is automatically achieved by charging the users.

1.3 The approach

One possible approach to stimulate a co-operative behavior and prevent congestion is to introduce the concept of money and service charges. The natural idea is that terminodes that used a service should be charged and terminodes that provided a service should be remunerated. To this end, we introduce a terminode currency that we call *beans*. We assume that the terminode hardware comes with an initial stock of beans. The terminode beans have no monetary value, and they can only be used within terminode networks.

Now, if a terminode wants to use a service (e.g., wants to send a message), then it has to *pay* for it in beans. This motivates each terminode to increase its number of beans, because beans are indispensable for using the network. Thus, the terminode is no longer interested in sending useless messages and overloading the network because this would decrease its number of beans, and it is better off providing services to other terminodes because this is the only way to earn beans[2].

1.4 Outline

In the sequel, we focus on the rewarding of one of the most important services that the terminodes should provide to each other, namely, packet forwarding. In Section 2, we introduce two approaches to solve this problem: the Packet Purse Model and the Packet Trade Model. The remaining sections are concerned with the implementation of these models. In Section 3, we summarize our general assumptions. Then, we present implementations that enforce the models in Section 4. Finally, in Section 5, we discuss the robustness and the efficiency of our solution, and, in Section 6, we conclude the paper.

2 Rewarding the packet forwarding service

2.1 The Packet Purse Model (PPM)

In this model, the originator of the packet pays for the packet forwarding service. The service charge is distributed among the forwarding terminodes in the following way: When sending the packet, the originator loads it with a number of beans sufficient to reach the destination. Each forwarding terminode acquires one or several beans from the packet and thus, increases the stock of its beans; the number of beans depends on the direct connection on which the packet is forwarded (long distance requires more beans). If a packet does not have enough beans to be forwarded, then it is discarded.

Packet forwarding in the Packet Purse Model is illustrated in Figure 1. Let us assume that originally each terminode has 7 beans (1). Furthermore, let us assume that A wants to send a packet to D. In order to do so, A loads, say, 5 beans in the packet and sends it to the next hop B (2). B takes out 1 bean from the packet, and forwards it with the remaining 4 beans to C (3). C takes out 2 beans from the packet and forwards it with the remaining 2 beans to the final destination D (4). Note that terminodes B and C, which forwarded the packet, increased their stock of beans, whereas terminode A, which originated the packet, decreased its stock of beans.

The basic problem with this approach is that it might be difficult to estimate the number of beans that are required to reach a given destination. If the originator under-estimates this number, then the packet will be discarded, and the originator loses its investment in this packet. If the originator over-estimates the number (like in our example above), then the packet will arrive, but the originator still loses the remaining beans in the packet[3]. The model described in the next subsection overcomes this problem.

[2]Similar to money in real life, beans can be lost as well. This loss has to be compensated somehow, otherwise the system gets poorer and poorer. One way to solve this problem is to let users buy beans. This would mean that providing services is, actually, not the only way to earn beans. However, it can be made the preferred way by appropriately choosing the price of one bean.

[3]Although, if the destination of the packet is a terminode that provides information services, then the remaining beans can be used to pay for these.

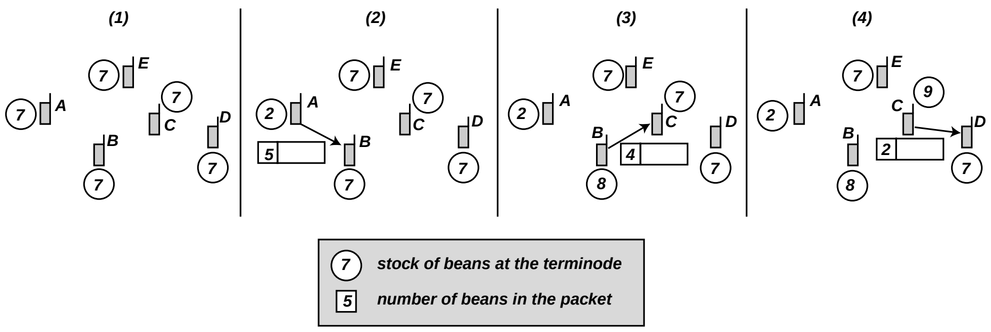
(1)
(2)
(3)
(4)
A
B
C
D
E
stock of beans at the terminode
number of beans in the packet

Figure 1: The Packet Purse Model

2.2 The Packet Trade Model (PTM)

In this approach, the packet does not carry beans, but it is traded for beans by intermediate terminodes. Each intermediary "buys" it from the previous one for some beans[4], and "sells" it to the next one (or to the destination) for more beans. In this way, each intermediary that provided a service by forwarding the packet, increases its number of beans, and the total cost of forwarding the packet is covered by the destination of the packet.

As an example, let us consider Figure 2. Let us assume that originally each terminode has 7 beans (1). Furthermore, let us assume that A wants to send a packet to D. A sends the packet to the first hop B for free (2). B then sells it to the next hop C for 1 bean (3). Finally, C sells it to the final destination D for 2 beans (4). Note that terminodes B and C, which forwarded the packet, increased their number of beans, whereas the destination D decreased its number of beans.

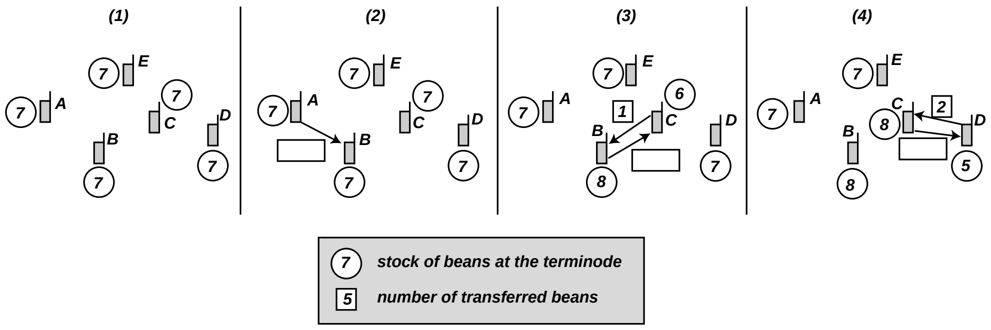

Figure 2: The Packet Trade Model

An advantage of this approach is that the originator does not have to know in advance the number of beans required to deliver a packet. Furthermore, letting the destination pay for the packet forwarding makes this approach applicable in case of multicast packets as well.

A disadvantage is that this approach for charging does not directly deter users from flooding the network. However, allowing each terminode to decide if it buys a packet or not can provide a sort of "back pressure" mechanism, which may deter a user from generating

[4]Except for the first intermediary that receives the packet for free from the originator.

too much traffic, by ensuring that eventually nobody will buy packets from users who try to overload the network.

2.3 Problems to be solved

Clearly, the models described above must be enforced somehow, otherwise the terminodes may depart from them. Terminodes (users) may misbehave in several ways if no enforcement and no protection are applied. One important general problem is, for instance, to prevent bean forgery.

In addition, the problems that we have to cope with in the Packet Purse Model include the following:

- The originator of a packet should be denied the re-use of the beans that it loaded in the packet purse.
- A forwarding terminode should be denied taking more beans out of the packet than it deserves for the packet forwarding (i.e., "packet robbery" should be prevented).
- Each intermediary should be forced to indeed forward the packet after having taken the beans out of it.
- The integrity of the packet purse should be protected during transit.
- The replay of a packet purse should be detected[5].
- Detachment of a packet purse from its original packet and re-use of it with another packet should be impossible.

Problems to be solved in the Packet Trade Model include the following:

- Each terminode should be denied the re-use of the beans that it spent for buying packets.
- A forwarding terminode should receive the beans from the next hop if, and only if, the next hop receives the packet from the forwarding terminode (fairness of the exchange).
- An intermediary should be prevented from selling the same packet several times (possibly to different next hops).

Furthermore, all the problems above should be solved in an efficient way; forwarding a single packet should not require complex cryptographic protocols and heavy computational effort, because the cost of these may well exceed the value of the service. We believe that we have found the best trade-off between robustness and efficiency in our implementations of the Packet Purse Model and the Packet Trade Model, which we present in the following sections.

[5]Consider the following subtle replay attack. An intermediary receives a packet with a packet purse, it copies them and then, simulates the reception of the same packet with the same packet purse several times (each time increasing its stock of beans) without forwarding the packet. If this kind of replay was not detected, then the intermediary can, actually, become arbitrarily rich from this single packet.

3 General assumptions

In this section, we summarize our general assumptions, which our implementations of the models described above rely on.

- **Tamper resistant security module.** We assume that each terminode has a tamper resistant security module, such as, for instance, a special chip or a smart card, that is used for the management of cryptographic parameters (e.g., keys) and beans. We assume that this security module functions correctly and its behavior cannot be modified by the user of the terminode or other attackers. Contrary to the security module, other parts of the terminode hardware and software are not tamper resistant and their behavior can be modified by anybody who has physical access to the device. We understand that regular users usually do not have the required level of knowledge and skills to modify their terminodes. Criminal organizations, however, can have enough interest and resources to reverse engineer a terminode and sell tampered terminodes with modified behavior on a large scale. Users may be interested in buying these tampered devices if they offer advantages over correctly behaving ones (e.g., longer battery lifetime). Our design goal is to distribute the terminode functions between the tamper resistant security module and the rest of the terminode device, which can be altered by an attacker, in a way that modification of the latter cannot give any advantages to the attacker.

- **Public key infrastructure.** We assume that there exists a public key infrastructure that the terminodes (or, more precisely, their security modules) can use to authenticate each other and to establish secure communication links. The design of an appropriate public key infrastructure for terminodes is an interesting and non-trivial problem that is beyond the scope of this paper. An approach to solve this problem is described in [5], other possible approaches are mentioned in [3].

- **Slow mobility.** We assume that the terminodes move "slowly" compared to the amount of traffic that goes through them. This is not to say that the terminodes must be physically slow, but that the neighborhood of a terminode does not change very fast. This makes it feasible for the terminode to keep track of its neighbors by running a sort of "hello protocol" at regular time intervals. Besides discovering its neighbors, the security module of the terminode uses the hello protocol to establish shared secrets with the security modules of its neighbors (different secrets with different neighbors, of course). The establishment of the shared secret is based on public key cryptography and relies on the existing public key infrastructure. In addition to the shared secret, we require that the security module agrees on the initial values of two counters with each of its neighbors. The shared secret and the two counters are used to protect the communication between neighboring security modules and will be discussed further in Section 4.

- **Omnidirectional antennae.** We assume that the terminodes use omnidirectional antennae, which means that a message sent by a terminode can be heard by all the terminodes within the communication range of the sender. We further assume that such a message can not only be heard, but it is understood by all of the neighbors. By this, we mean that all the neighbors receive the message and can determine who the sender and the intended receiver are and what the content of the message is[6]. Depending on the MAC layer used, this may require that the terminodes agree on further parameters with their neighbors during the hello protocol. If, for instance, access to the shared radio resource is based on code division (CDMA), then the terminode should inform its neighbors about all the codes that it uses, in order for the neighbors to be able to receive messages sent by the terminode.

- **Reliable communication between neighbors.** For the sake of simplicity, we assume that the communication between neighboring terminodes is reliable. This means that if a message is sent successfully (e.g., without any collision), then it arrives to the intended next hop correctly. We will address the problem of unreliable communication links in a future paper. We note, however, that this assumption does not imply that end-to-end communication is reliable. Since messages can be modified and intercepted by the forwarding terminodes themselves, successfully sending a message to the next hop does not mean that the message will correctly arrive to the final destination.

- **Pricing.** In the Packet Purse Model, we assume that there exists a mechanism to estimate the number of beans that the originator of a packet must load in the packet purse in order for the packet to be delivered to the final destination. Furthermore, we also assume that there is a mechanism to determine the number of beans that a forwarding terminode can acquire from a packet purse. Similarly, in the Packet Trade Model, we assume that there exists a mechanism to determine the number of beans, for which a forwarding terminode can sell a packet to the next hop.

  In order to ease presentation, in this paper, we assume that each forwarding terminode should be rewarded with exactly one bean for the packet forwarding. This means, that in the Packet Purse Model, each intermediate terminode that forwards the packet can take exactly one bean out of it, and in the Packet Trade Model, each forwarding intermediary can sell the packet for one more bean than it paid for. Our solution, however, works without modifications in the general case as well.

- **Terminodes are greedy.** We assume that terminodes are greedy, and they always want to increase their number of beans. On one hand, this is reasonable, because beans are indispensable for using the network. On the other hand, there might be situations, where greediness is not the best strategy. Consider, for instance, a terminode that has a lot of beans, but whose battery is almost exhausted. In this situation, earning more beans has clearly less benefit, than saving battery power. But if the terminode is greedy, then it keeps on forwarding packets, and uses up all of its energy. It would be more realistic to assume that the behavior of the terminode depends on both the number of its beans and the status of its battery. This issue is left for further study.

- **No network operator.** We assume that the network is totally self-organized and self-operated. Users simply purchase terminodes and use them. The inter-working with existing fixed and wireless networks is left for future study.

[6]More precisely, each neighbor can see the bits of the message, although not necessarily understanding the real meaning of the message (e.g., in case of end-to-end encrypted messages).

4 Implementing the models

We use the tamper resistant security module to enforce the behavior described by the models. In this section, we present the description of this module and the protocols that it runs with its environment. Our leading design principle is to put as little as possible in the security module in order to rely on as few assumptions as possible.

4.1 Long and medium term data in the security module

The security module stores and manipulates data that is critical for the correct behavior of the system. Since the security module is tamper resistant, this data cannot be corrupted by the user of the terminode or other attackers.

The following long term data are stored in the security module SM:

- **Unique identifier.** The security module stores its system-wide unique identifier, which we denote by id_{SM}.
- **Private key.** The security module has a public key and a corresponding private key. The private key is exclusively known to SM and, thus, it must be stored by SM. The public key does not need to be kept secret, therefore, it can be stored elsewhere. It is important, however, that other security modules associate the right public key (i.e., the public key of SM) with the unique identifier of SM. This is ensured with the help of the assumed public key infrastructure.
- **Number of beans.** Beans are represented by counters in the security modules of the terminodes. The wealth of each terminode is equal to the value of the bean counter in its security module. We denote the bean counter in the security module by b_{SM}.

In addition, the security module keeps a list of current neighbors and maintains data associated to each of these. SM stores the following medium term data for each neighboring security module SM':

- **Unique identifier.** The system-wide unique identifier $id_{SM'}$ of the neighbor.
- **Shared secret key.** When SM and SM' become neighbors, they establish a shared secret key $k_{SM,SM'}$ between them using the hello protocol and public key cryptography. This shared secret is exclusively known to SM and SM', and it is used to protect the communication between them. This protection, in turn, is based on symmetric key cryptography for efficiency reasons. Protection is necessary, because the security modules cannot communicate directly but only through their hosting terminodes, which are under the control of (potentially malicious) users.
- **Sending and receiving counters.** SM stores a sending counter $n_{SM \to SM'}$ and a receiving counter $n_{SM \leftarrow SM'}$ associated with SM'. These counters are used to detect message replay, which, as mentioned in Subsection 2.3, would fool the security module to process the same message twice. SM' has similar counters $n_{SM' \to SM}$ and $n_{SM' \leftarrow SM}$, which are associated with SM. When SM and SM' become neighbors, they initialize their receiving counters to random values and use the hello protocol to set their sending counters such that the following holds: $n_{SM \to SM'} = n_{SM' \leftarrow SM} + 1$ and $n_{SM' \to SM} =$

$n_{SM \leftarrow SM'} + 1$. Then, each time SM sends a message to SM', it includes the current value of its sending counter $n_{SM \rightarrow SM'}$ in the message, and then increments the counter. When SM receives a message from SM', it verifies if the message contains a counter value that is greater than its current receiving counter $n_{SM \leftarrow SM'}$. If so, then it accepts the message and increases its counter to the received value, otherwise it rejects the message. SM' behaves similarly.

- **Fine.** Another counter is $f_{SM,SM'}$, the initial value of which is 0. SM uses this counter to account for the misbehavior of the terminode that hosts SM' with respect to the terminode that hosts SM. The protocols that are used by the security modules are such that SM does not immediately increase its bean counter if its hosting terminode forwarded a packet, but it waits for an acknowledgment from the security module SM' of the next hop in order to be sure that the packet has indeed been forwarded. If this acknowledgement does not arrive, then SM records the misbehavior of the next hop by increasing the fine counter $f_{SM,SM'}$ associated with SM'. The next time it sends a packet to the same next hop, SM also sends the value of the fine counter. If this packet is processed by the next hop, then SM' takes into account the fine by decreasing its bean counter accordingly, and SM can reset its fine counter. If, however, this packet is not processed either (i.e., no acknowledgement arrives), then SM further increases the fine counter. If the counter exceeds a limit, then the hosting terminode of SM may stop forwarding packets toward the misbehaving next hop. This mechanism stimulates terminodes to send acknowledgements.

  We should note that a missing acknowledgment does not necessarily mean that the next hop is misbehaving and did not send it. It is also possible that the hosting terminode of SM cheated and it did not actually forward the packet or it falsely claims the acknowledgement to be missing. However, we assume that this is not the case, because it would contradict our assumption about the greediness of the terminode: the terminode cannot increase its number of beans by not forwarding the packet or claiming an arrived acknowledgement missing, whereas it can increase its number of beans if it behaves correctly.

4.2 Implementing the Packet Purse Model

4.2.1 The Packet Purse Header (PPH)

In the Packet Purse Model, each packet has to carry some beans required to forward the packet. These beans are stored in the Packet Purse Header (PPH), which is an additional header between the MAC Layer Header and the Network Layer Header as it is illustrated in Figure 3. The PPH is created and manipulated by security modules. It is cryptographically protected in order to prevent forgery and illegitimate modification during transit.

The PPH is re-computed by the security module of each forwarding terminode. It has three parts: a part that is intended for the security module of the next hop, another part that is an acknowledgement for the security module of the previous hop, and a third one that is common and intended for both the next and the previous hops. The common part contains only the unique identifier of the security module that computed this PPH. The acknowledgement part contains the identifier of the security module of the previous hop, the sending counter that was received from that hop, and an Acknowledgement Authentication Code (AAC) that is computed from the previous PPH, which was attached to the packet,

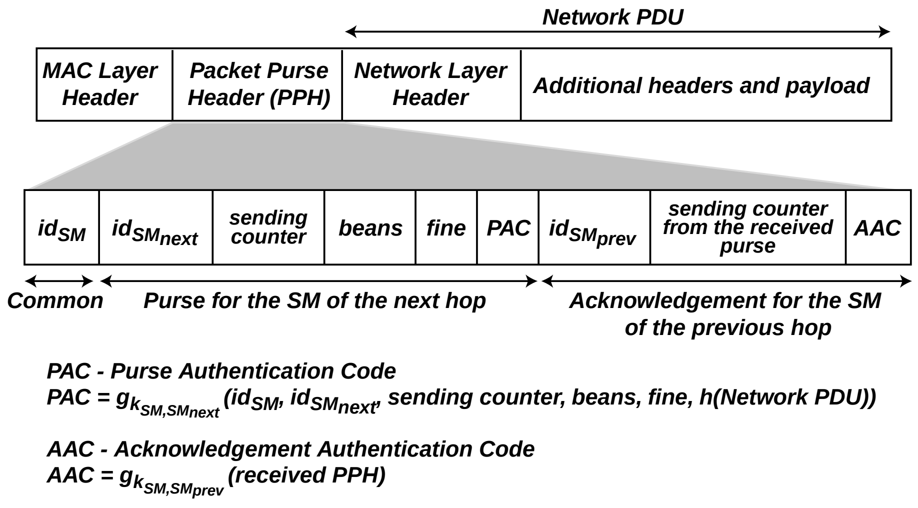
Network PDU
MAC Layer Header
Packet Purse Header (PPH)
Network Layer Header
Additional headers and payload
id_SM
id_SMnext
sending counter
beans
fine
PAC
id_SMprev
sending counter from the received purse
AAC
Common
Purse for the SM of the next hop
Acknowledgement for the SM of the previous hop
PAC - Purse Authentication Code
PAC = g_k_SM,SMnext(id_SM, id_SMnext, sending counter, beans, fine, h(Network PDU))
AAC - Acknowledgement Authentication Code
AAC = g_k_SM,SMprev(received PPH)

Figure 3: The Packet Purse Header (PPH)

using a keyed cryptographic hash function g, where the key is the shared secret between this security module and the security module of the previous hop. Finally, the purse part that is intended for the security module of the next hop contains the identifier of that security module, the sending counter associated with that security module, the number of beans in the packet, a fine to be paid by the next hop, and a Purse Authentication Code (PAC), which is computed from the purse part of the PPH and the cryptographic hash value $h(NetworkPDU)$ of the content of the packet using a keyed cryptographic hash function g, where the key is the shared secret between this security module and the security module of the next hop.

As it can be seen from the description, the acknowledgement that is intended for the previous hop is *piggy backed* on the packet that is sent to the next hop. Here, we rely on the assumption that the terminodes have omnidirectional antennae, and they receive all messages that are emitted by neighboring terminodes. Thus, when a terminode forwards a packet to the next hop, the previous hop, from which this packet has arrived, also receives it, and extracts the acknowledgement.

4.2.2 The packet forwarding protocol

The packet forwarding protocol is illustrated in Figure 4, where we assume that terminode T_q has received a packet from terminode T_p (which received it from the previous hop T_o), and T_q wants to forward it to T_r. To do so, T_q has to obtain a new Packet Purse Header PPH' from its security module SM_q by supplying it with the identifier of the security module of the next hop, the Packet Purse Header PPH received from the previous hop, and the cryptographic hash value $h(NetworkPDU)$ of the content of the packet.

SM_q first verifies PPH. It reads the identifier of its sender SM_p from the common part of PPH. Then, it verifies if the sending counter in PPH is greater than the receiving counter $n_{q\leftarrow p}$ associated with SM_p. If so, then this PPH is not a replay (i.e., it has not yet been processed by SM_q), and SM_q proceeds by setting $n_{q\leftarrow p}$ to the received counter value. SM_q then verifies the authenticity of PPH by re-computing the Purse Authentication Code and comparing the computed value to the received one. If they match, then it knows that PPH

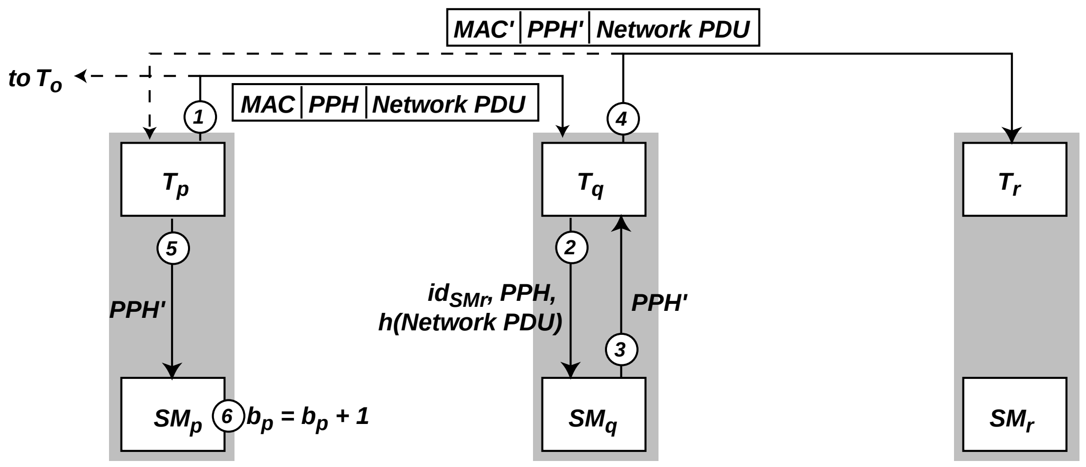

Figure 4: The packet forwarding protocol

has indeed been created by SM_p and has not been modified . Finally, it checks if there is a fine to be paid, and if so, then it decreases its bean counter accordingly.

After successful verification, SM_q calculates the new Packet Purse Header PPH'. This is illustrated in Figure 5. It puts its own identifier id_{SM_q} in the common part. It decreases the number of beans in the packet by one, and constructs the purse part by including the identifier of the next security module SM_r, the sending counter $n_{q \to r}$ associated with SM_r, the number of beans in the packet, the fine counter $f_{q,r}$ associated with the next hop, and the Purse Authentication Code $PAC_{q,r}$ calculated from the purse and the hash value of the content of the packet using the cryptographic hash function g and the shared secret $k_{q,r}$. Then SM_q increases its sending counter $n_{q \to r}$, and constructs the acknowledgement part by including the identifier of SM_p, the sending counter $n_{p \to q}$ form the purse part of PPH, and the Acknowledgement Authentication Code $AAC_{q,p}$, which is calculated from PPH using the cryptographic hash function g and the shared secret $k_{q,p}$. Finally, SM_q stores PPH' internally, and outputs a copy for T_q.

T_q attaches the new Packet Purse Header PPH' to the packet and sends it to T_r. T_p also receives the forwarded message, and it can recognize that there is an acknowledgement for its security module SM_p in the packet, because PPH' contains the identifier of SM_p in the acknowledgement part. T_p uploads PPH' to its security module. SM_p tries to find PPH in its internal memory by matching the identifier of SM_q and the sending counter received in the acknowledgement part of PPH' to the identifiers and sending counters in the purse part of stored pending Packet Purse Headers. If SM_p finds PPH, then it verifies the authenticity of the acknowledgement in PPH' by re-computing the Acknowledgement Authentication Code from PPH and comparing it to the value received in PPH'. If they are equal, then SM_p increases its bean counter by one, decreases its fine counter $f_{p,q}$ by the the value of the fine in PPH (but never lets it become less than 0), and deletes PPH from its internal memory.

T_p keeps track of the forwarded but not yet acknowledged packets. If no acknowledgement arrives to a packet after a given time, then T_p notifies its security module, which increases the fine counter that is associated with the misbehaving neighbor and deletes the corresponding Packet Purse Header from its internal memory. Although it would be simpler if the security module itself measured the time-out, we do not want to require the security module to have

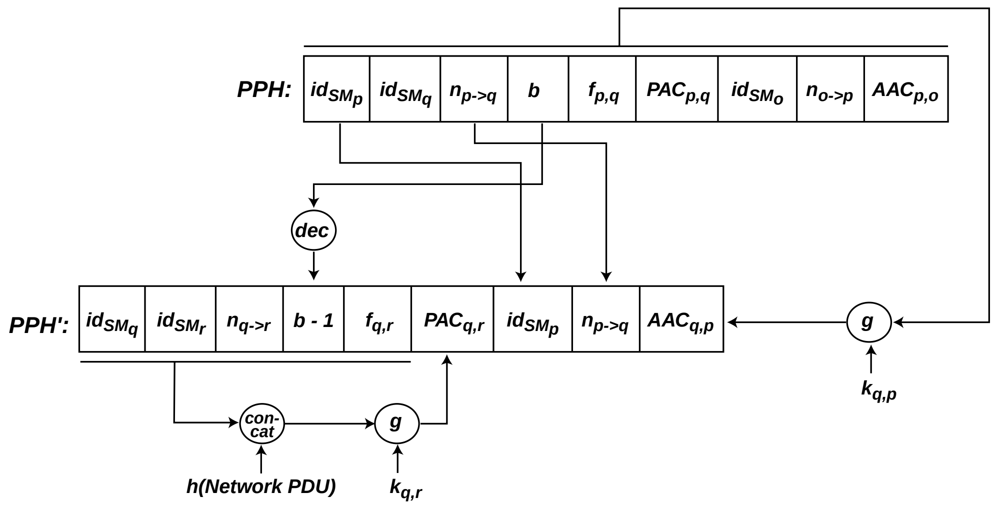

Figure 5: Re-computing the PPH

an internal clock, because this would also require an internal source of energy, and we believe that building such a tamper resistant module is quite difficult. Our solution still works well, because T_p is not interested in signaling a missing acknowledgement if the acknowledgement has indeed arrived: it can increase its number of beans by uploading the acknowledgement, while it cannot gain anything by claiming it missing.

4.2.3 Packet creation and final delivery

Before the packet is sent by its originator, the security module decreases its bean counter by the number of beans specified by the originator and creates a PPH that contains the same number of beans. This PPH is a special one, because it does not have any acknowledgement part, since there is no previous hop that would need it.

When the packet is delivered to its final destination, then the PPH is loaded in the security module, which creates a special PPH' that has only an acknowledgement part. The destination should send an empty packet with this special PPH' to the previous hop. If it does not send it, then the security module of the previous hop increases its fine counter associated with the destination, and the destination will be punished for the misbehavior later.

4.3 Implementing the Packet Trade Model

The Packet Trade Model can be implemented in the same way as the Packet Purse Model. Like before, each packet has an additional header, which we call Packet Trade Header (PTH). The structure of the PTH is the same as the structure of the PPH, with the only difference that instead of the number of beans, it contains the price of the packet. The same packet forwarding protocol described before applies in the Packet Trade Model as well with a minor modification. Now, the security module of each forwarding terminode decreases its bean counter by the price in the PTH (buying) and increases the price by one when re-computing the PTH, and increases its bean counter by the new price when the acknowledgement arrives (selling).

5 Analysis

In this section, we shortly analyze the implementation of the Packet Purse Model described above. We show how the implementation solves our original problems of stimulation for co-operation and prevention of overloading, and discuss its robustness and efficiency. Essentially, this analysis applies for the implementation of the Packet Trade Model as well, since it is almost identical to the implementation of the Packet Purse Model. We will point out those cases in which the analysis does not apply for the Packet Trade Model.

5.1 Stimulation for co-operation and prevention of overloading

Our implementation encourages users to keep their terminodes switched on and let them forward packets, because this is the only way to increase their number of beans. If a terminode does not forward a packet, then it will receive a fine later, and its number of beans will be decreased. In addition, if a terminode denies packet forwarding for a long time, then no more packet will be sent to it.

Our implementation of the Packet Purse Model discourages users to send useless traffic and overload the network because this would decrease their number of beans. Our solution ensures that the benefit each user gets from the network does not exceed what she contributes to it.

We should note, however, that our implementation of the Packet Trade Model does not deter users from overloading the network. The reason is that, contrary to the original idea of the Packet Trade Model, our implementation does not allow a terminode to decide whether it buys a packet or not. Instead, a terminode is forced to buy each packet that is sent to it. This means that any terminode can generate useless traffic and overload the network without any consequences. In order to solve this problem, our implementation must be modified to allow each terminode to decide whether to buy a packet or not. This would provide a sort of "back pressure" mechanism, which may ensure that eventually nobody will buy packets from misbehaving senders. This issue is left for further study.

5.2 Robustness

The implementation described above is robust and resists against various attacks. Bean forgery is prevented, because it would require either an illegitimate increase of the bean counter, or the generation of fake packet purses or acknowledgements. The former is impossible, because the bean counter is manipulated by the security module, which functions correctly and its behavior cannot be altered. The latter is prevented by the use of cryptographic checksums (i.e., the Purse Authentication Code and the Acknowledgement Authentication Code), which can be computed correctly only by the security module. These checksums also protect the integrity of the PPH during transit. Furthermore, the packet purse cannot be detached from the packet and re-used with another one, because the calculation of the Purse Authentication Code involves the cryptographic hash value of the content of the packet. Replay of the packet purse is prevented by the use of an ever increasing counter that is placed in the purse. This solution is preferable to the application of time-stamps, because it does not require the security module to have an internal clock and to run clock synchronization protocols.

The originator of a packet cannot re-use the beans that it has already loaded in the packet, because the security module decreases the bean counter when creating a PPH for a new packet. An intermediary cannot take out more beans from the packet than it deserves for the packet forwarding, because its bean counter can be manipulated exclusively by its security module, which behaves correctly. Moreover, the intermediary is stimulated to forward the packet, because its bean counter will be increased only if an acknowledgment arrives from the next hop, and this is only possible if the packet has been forwarded.

Our solution requires each hop to send an acknowledgement for the packet it received. Terminodes, however, may be reluctant to send acknowledgements, because sending consumes energy and it does not have any direct advantages. This problem is related to fair exchange [6, 7] (in our case, packets for acknowledgements), and it is usually solved with the involvement of a trusted third party (TTP). We cannot, however, assume the existence of TTPs in terminode networks. The problem of fair exchange without a TTP is analyzed in [8], where it is called *unenforced safe exchange*. The author proves that isolated unenforced safe exchange is not possible if the last step of the exchange has some costs. A proposed solution is that one should not consider only a single isolated exchange, but one should also take into account possible future exchanges, where the behavior of the parties in the future exchanges may depend on the result of the current exchange. If misbehavior in the present can be punished in the future, then unenforced safe exchange becomes possible. In our implementation, we used these ideas in two ways to stimulate terminodes to send acknowledgments. First, we reduced the cost of sending an acknowledgement by piggy backing it to a normal packet that the terminode sends anyway (except for the destination of a packet). Second, we introduced fines, in order to punish misbehaving terminodes. Moreover, the fine is sent in the purse together with the beans, which enforces the terminode who wants the beans to upload the fine as well to the security module that will decrease the bean counter according to the received fine.

We should note that exchanges without TTP can never achieve the same level of fairness as those with TTP. The existence of different levels of fairness is discussed in [9], where the authors relate the different levels to different equilibrium concepts in game theory. According to these results, our implementation achieves Nash-equilibrium fairness, which essentially means that a misbehaving party may cause some damage to a correctly behaving one, but it also loses something or at least cannot gain anything (apart from malicious joy) with the misbehavior.

5.3 Efficiency

At first sight, our solution may seem a bit heavy to implement. However, the overhead generated by it is small when compared to all the functions that are required to accomplish packet forwarding. In particular, the calculation and verification of the Packet Purse and the Packet Trade Headers require only cryptographic hash function computations, which can be done very efficiently [10]. Public key cryptographic operations are used only rarely (in the hello protocol). Moreover, most of the processing load will be supported by the security module; to some extent, it can be accomplished in parallel with the processing performed by the main processor of the terminode.

Another issue is the length of the Packet Purse Header. Assuming that the identifiers of the security modules are 8 byte long, the sending and receiving counters are 6 byte long, the Purse and the Acknowledgement Authentication Codes are 20 byte long, and the beans and the fine are both represented on 2 bytes, we get that the Packet Purse Header is 80 byte long. We cannot further assess whether this is acceptable or too much, because of the lack of information about the length of other headers and the average length of the packets.

Efficiency can be improved by using the Packet Purse Header and all the related mechanisms only in a small fraction of packets. Then the majority of the packets would not carry an additional header and would be processed without any call to the security module. This means, however, that the terminodes would not be rewarded for the forwarding of each packet, and we would have to ensure that they forward those packets as well from which they cannot expect any beans. This issue is left for future work.

6 Conclusion

In this paper, we addressed the problem of service availability in terminode networks (mobile ad-hoc WANs). We have presented a secure mechanism to stimulate end users to keep their terminodes turned on, to refrain from overloading the network, and to thwart tampering aimed at converting the device into a "selfish" one.

This work was motivated by the experience of cellular networks, which has proven that as soon as mobile stations are under the control of end users, there is a strong temptation to alter their behavior in one way or another. Therefore, all facets of security have to be carefully analyzed and implemented. We are currently working on the integration of the proposed solution with other security functions, such as confidentiality and integrity protection of communications.

Finally, we believe that introducing a kind of virtual currency can serve several other purposes in mobile ad-hoc WANs. First, it can be used to remunerate not only communication services, as described in this paper, but also information services. Second, it can be used as a way to pay for the usage of backbones or satellite links, when a terminode has to communicate with a very distant party. In this case, the virtual currency will have to be converted in some way into "hard" currency.

7 Acknowledgement

We are thankful to Refik Molva for proposing the use of ever increasing sending and receiving counters instead of time-stamps to detect replays.

References

[1] J.-P. Hubaux, J.-Y. Le Boudec, S. Giordano, and M. Hamdi. The Terminode project: toward mobile ad-hoc WANs. In *Proceedings of the Mobile Multimedia Conference, MOMUC*, San Diego, 1999.

[2] Terminodes web site. `http://www.terminodes.org/`.

[3] J.-P. Hubaux, J.-Y. Le Boudec, M. Vojnović, S. Giordano, M. Hamdi, L. Blazević, and L. Buttyán. Toward mobile ad-hoc WANs: Terminodes. Technical Report No. DSC/2000/006, Swiss Federal Institute of Technology, Lausanne, January 2000.

[4] F. Stajano and R. Anderson. The resurrecting duckling: security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, Cambridge, UK, April 1999.

[5] L. Zhou and Z. Haas. Securing ad-hoc networks. *IEEE Network*, 13(6):24–30, November-December 1999.

[6] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 55–61, May 1996.

[7] M. Franklin and M. Reiter. Fair exchange with a semi-trusted third party. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 1–6, April 1997.

[8] T. Sandholm. Unenforced e-commerce transactions. *IEEE Internet Computing*, 1(6):47–54, November-December 1997.

[9] L. Buttyán and J.-P. Hubaux. Toward a formal model of fair exchange – a game theoretic approach. Technical Report No. SSC/1999/039, Swiss Federal Institute of Technology, Lausanne, December 1999.

[10] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1997.